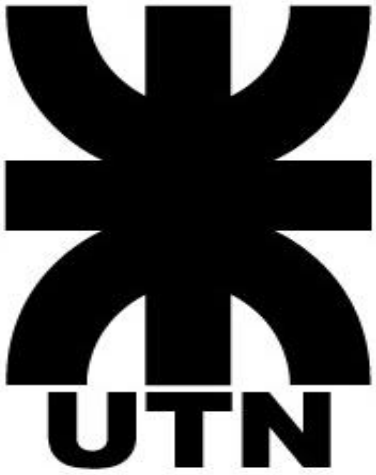

Universidad Tecnológica Nacional
Facultad Regional Concepción del Uruguay

Ingeniería en Sistemas de Información

Sistemas Operativos

Trabajo Práctico Final:
"Ampliación y mejora del shell experimental"

Integrantes del Grupo:

Arrúa, Martin
Lazbal, David
Rivera, Ramiro

Fecha de Entrega:
19/02/2015

Docentes:

Ing. Arellano Gabriel
Ing. Aguiar Osvaldo

```
UNIT analizador;

INTERFACE

      Uses
            errors, ALR, comandos, utilidades, unix, baseunix, sysutils;

function analizar       (var ENTRADA: string): boolean;          // Analiza la
cadena introducida por el usuario.
function analizarCD           (DIRECTORIO: string): cint;              // Ana-
liza la cadena para ejecutar el comando CD.
function analizarEXEC   (var ENTRADA:string): cint;              // Analiza la
cadena para ejecutar el comando EXEC.
procedure analizarCAT   (var ENTRADA: string);                   // Analiza la
cadena para ejecutar el comando CAT.
procedure analizarKILL  (var ENTRADA: string);                   // Analiza la
cadena para ejecutar el comando KILL.
procedure analizarLS    (var ENTRADA: string);                   // Analiza la
cadena para ejecutar el comando LS.
procedure analizarPWD   (var ENTRADA: string);                   // Analiza la
cadena para ejecutar el comando PWD.
procedure analizarReDIR (var ENTRADA: string);                   // Analiza la
cadena si presenta redirección de la salida estándar.
procedure analizarPipe  (var ENTRADA: string);                   // Analiza la
cadena si presenta pipes (tuberías).
procedure analizarBG    (var ENTRADA: string);                   // Analiza la
cadena si presenta solicitud de ejecutarse en 2do plano.


IMPLEMENTATION

function analizar(var ENTRADA: string): boolean;
var COMANDO, ARGUMENTOS: string;
Begin
      analizar := true;
      COMANDO     := extraerComando(ENTRADA);              //se extrae el co-
mando para el case
      ARGUMENTOS:= ENTRADA;                                //se obtienen los ar-
gumentos del comando
      if ARGUMENTOS <> '' then           //se vuelve a la entrada original
            ENTRADA := COMANDO + ' ' + ARGUMENTOS
      ELSE
            ENTRADA := COMANDO;
      COMANDO     :=upcase(COMANDO);
      if COMANDO='EXIT' then analizar:= false        // Salir del programa.
      else
            begin
            if COMANDO='JOBS' then mostrarTabla            // Muestra las
tareas.
            else
                  begin
                  if COMANDO='HELP' then help              // Muestra una panta-
lla de ayuda.
                  else
                        begin
                        if solicitudBG(COMANDO) then bg(ARGUMENTOS)
      // Solicita ejecutar un proceso en segundo plano.
                        else
                              begin
```

```
                        if solicitudFG(COMANDO) then fg(ARGUMENTOS)
          // Solicita ejecujar un proceso en primer plano.
                        else begin
                              if enBg(ENTRADA) then begin analizarBG(EN-
TRADA); writeln('esto dice que puse en &') end       // Envía al segundo plano.
                              else begin
                                    if    redireccion(ENTRADA) then anali-
zarReDIR(ENTRADA)       // Redirección de la salida estándar.
                                    else begin
                                          if tuberia(ENTRADA) then anali-
zarPipe(ENTRADA)              // Tubería (pipe).
                                          else begin
                                                if COMANDO='CAT' then ana-
lizarCAT(ARGUMENTOS)          // Comando CAT
                                                else begin
                                                      if COMANDO='CD' then
analizarCD(ARGUMENTOS)  // Comando CD
                                                      else begin
                                                            if CO-
MANDO='KILL' then analizarKILL(ARGUMENTOS)      // Comando KILL
                                                            else begin
                                                                  if CO-
MANDO='LS' then analizarLS(ARGUMENTOS)    // Comando LS
                                                                  else
begin
                                                                        if
COMANDO='PWD' then analizarPWD(ENTRADA)   //Comando PWD
      else
      begin
      if COMANDO='MOO' then moo(ARGUMENTOS)
      else
            begin
            if ENTRADA <> '' then         // Comando externo
                  Begin
                  if analizarEXEC(ENTRADA) = -1 then
                        begin
                        mostrarerror(5);
                        exit;
                  end;                    //Comando no reconocido.
            end;
      end;
      end;
                                                                  end;
                                                            end;
```

```
                                                                  end;
                                                            end;
                                                      end;
                                                end;
                                          end;
                                    end;
                              end;
                        end;
                  end;
            End;
end;


procedure analizarCAT(var ENTRADA: string);
var dir1, dir2: string;
      tipo: byte;
Begin
      if strOutPut = '' then tipo := 0 else tipo := 1; // Tipo 0: Salida están-
dar, tipo 1: Redirigir la salida.
      dir1 := extraerComando(ENTRADA);                    // se hace reuso de
esta funcion debido al analisis sintáctico, se obtiene el directorio del primer
archivo
      if dir1 = '' then
            begin
            mostrarerror(1);
            exit;
            end
      else
            begin
                  dir2 := ENTRADA;                              // se asigna a
dir2 la direccion del segundo archivo
                  if dir1[1] <> '/' then dir1 := dirActual+'/'+dir1;
                  if (dir2 <> '') and (dir2[1] <> '/') then dir2 := dirAc-
tual+'/'+dir2;
                  CAT (dir1,dir2,tipo);
            end;
End;



function analizarCD(DIRECTORIO: string): cint;
{
      Nombre: analizarCD.
      return: -1 si hay error, 0 si se ejecutó con éxito.
}
Begin
      analizarCD:= 0;
      if (DIRECTORIO='') then
            fpChDir(dirHome)
      else
            begin
                  if fpChDir(DIRECTORIO) <> 0 then
                        analizarCD:= -1;
            end;
End;



function analizarEXEC(var ENTRADA:string): cint;
```

```
{
   Nombre: analizarEXEC.
   Condición: No se permiten más de 3 parámetros.

}

var comando : string;
      aux         : parametros;
      tipo  : byte;
      i           : integer;
Begin
      analizarExec := 0;
      if (strOutPut = '') then tipo:= 0 else tipo:= 1;
            comando:=extraerComando(ENTRADA);         //ENTRADA vuelve sin el co-
mando.
            aux:=extraerArgumentos(ENTRADA,i);        //AUX se le asignan los pa-
rámetros de forma -* y ENTRADA contiene cualquier parametro sin la forma -*,
como podría ser un directorio
            if not(fileExists(comando)) then
                  comando:=FSearch(comando,strpas(fpGetenv('PATH')));
            if comando <> '' then
                  begin
                        if (ENTRADA = '') then        // Sólo existen parámetros
de la forma -*.
                              begin
                                    case i of   // 0, 1, 2 o 3 parámetros.
                                          0: EXEC(comando,[],tipo);
                                          1: EXEC(comando,[aux[1]], tipo);
                                          2: EXEC(comando,[aux[1],aux[2]],
tipo);
                                          3: EXEC(co-
mando,[aux[1],aux[2],aux[3]], tipo);
                                    end;
                              end
                        else                          // Existe algún parámetro
sin la forma -*, como podría ser un directorio
                              begin
                                    case i of   // 0, 1, 2 o 3 parámetros + algún
directorio o parámetro extra..
                                          0: EXEC(comando,[ENTRADA], tipo);
                                          1: EXEC(comando,[aux[1],ENTRADA],
tipo);
                                          2: EXEC(comando,[aux[1],aux[2],EN-
TRADA], tipo);
                                          3: EXEC(co-
mando,[aux[1],aux[2],aux[3],ENTRADA], tipo);
                                    end;
                              end;
                  end
            else
                  analizarExec:= -1;
End;



procedure analizarKILL(var ENTRADA: string);
{
   Nombre: analizarKILL.
```

```
   Condición: Deben pasarse dos parámetros, sin excepción. Forma: kill -n_signal
pid
}
var argumentos: parametros;
      proc,signal: longint;
      i: integer;
      err: word;
Begin
      argumentos:= extraerArgumentos(ENTRADA,i);                  //se separan la
señal y el pid recibidos (pid en entrada)
      if i <> 1 then
      //verifica que se pase sólo 1 señal
            begin
            mostrarerror(8);  exit;
            end
      else
      begin
            val(argumentos[1],signal,err);
      //se asigna a signal el numero de señal recibido
            if err<>0 then
                  begin
                  mostrarerror(9);  exit;
                  end
            else
                  begin
                        val(ENTRADA,proc,err);
      //se asigna a signal el numero de señal recibido
                        if err<>0 then
                              begin
                              mostrarerror(9);
                              exit;
                              end
                        else
                              KILL(signal,proc);
                  end;
      end;
End;

procedure analizarLS(var ENTRADA: string);
{
   Nombre: analizarLS.
   Opciones: Puede haber 0, 1, 2 o 3 parámetros identificados con un guión <->
                  y pueden estar en cualquier orden. Estos son <-a>, <-f>, <-
l>.
                  Puede haber o no 1 argumento que indique la ruta desde la
cual
                  trabajar, luego de los parámetros comenzados con guión.
}
var
      directorio,cad,dirBase  : string;
      i,j                              : integer;
      argumentos                 : parametros;

Begin
      dirBase:= dirActual;
      argumentos:= extraerArgumentos(ENTRADA,i);
      for j:=1 to i do
            argumentos[j]:=upcase(argumentos[j]);
      directorio:= ENTRADA;
```

```
      case i of
      0:          begin
                  if (directorio <> '') then
                        cad:= (directorio)            //si existe el directorio
se lo prepara para cd
                  else
                        cad:= (dirBase);        //si no existe se usará el direc-
torio actual
                  if analizarCD(cad) = 0 then               //se posiciona en el
directorio y verifica la salida
                        ls(false,false,false)               //sí = 0 se posicionó
bien y ejecuta ls
                  else
                        begin
                        mostrarerror(4);  exit;
                        end                                          //si <> 0 el
directorio recibido no es coherente.
                  end;

      1:          begin
                  if not(paramValido(argumentos[1])) then
                        begin
                              mostrarerror(8); exit;
                        end;

                  if (directorio <> '') then
                        cad:= (directorio)            //si existe el directorio
se lo prepara para cd
                  else
                        cad:= (dirBase);        //si no existe se usará el direc-
torio actual
                  if analizarCD(cad) = 0 then               //se posiciona en el
directorio y verifica la salida
                        ls(                                                  //si = 0
se posicionó bien y ejecuta ls
                              pos('A',argumentos[1])<>0,    //verifica si el ar-
gumento contiene el parametro 'A'
                              pos('F',argumentos[1])<>0,    //verifica si el ar-
gumento contiene el parametro 'F'
                              pos('L',argumentos[1])<>0)    //verifica si el ar-
gumento contiene el parametro 'L'
                  else
                        begin
                        mostrarerror(4);  exit;
                        end                                          //si <> 0 el
directorio recibido no es coherente.
                  end;

      2:          begin
                  if not(paramValido(argumentos[1])) or not(paramValido(argumen-
tos[2])) then
                        begin
                              mostrarerror(8); exit;
                        end;

                  if (directorio <> '') then
                        cad:= (directorio)            //si existe el directorio
se lo prepara para cd
                  else
```

```
                        cad:= (dirBase);        //si no existe se usará el direc-
torio actual
                  if analizarCD(cad) = 0 then               //se posiciona en el
directorio y verifica la salida
                        ls(                                             //sí = 0
se posicionó bien y ejecuta ls
                              ((pos('A',argumentos[1])<>0) or (pos('A',argumen-
tos[2])<>0)),     //verifica si alguno de los argumentos contiene el parametro
'A'
                              ((pos('F',argumentos[1])<>0) or (pos('F',argumen-
tos[2])<>0)),     //verifica si alguno de los argumentos contiene el parametro
'F'
                              ((pos('L',argumentos[1])<>0) or (pos('L',argumen-
tos[2])<>0)))     //verifica si alguno de los argumentos contiene el parametro
'L'
                  else begin
                        mostrarerror(4);
                        exit;
                        end;                            //si <> 0 el directorio re-
cibido no es coherente.
                  end;

      3:          begin
                  if not(paramValido(argumentos[1])) or not(paramValido(argumen-
tos[2])) or not(paramValido(argumentos[3])) then
                        begin
                              mostrarerror(8); exit;
                        end;

                  if (directorio <> '') then
                        cad:= (directorio)            //si existe el directorio
se lo prepara para cd
                  else
                        cad:= (dirBase);        //si no existe se usará el direc-
torio actual
                  if analizarCD(cad) = 0 then               //se posiciona en el
directorio y verifica la salida
                        ls(                                             //si = 0
se posicionó bien y ejecuta ls
                              ((pos('A',argumentos[1])<>0) or (pos('A',argumen-
tos[2])<>0) or (pos('A',argumentos[3])<>0)),    //verifica si alguno de los argu-
mentos contiene el parametro 'A'
                              ((pos('F',argumentos[1])<>0) or (pos('F',argumen-
tos[2])<>0) or (pos('F',argumentos[3])<>0)),    //verifica si alguno de los argu-
mentos contiene el parametro 'F'
                              ((pos('L',argumentos[1])<>0) or (pos('L',argumen-
tos[2])<>0) or (pos('L',argumentos[3])<>0)))    //verifica si alguno de los argu-
mentos contiene el parametro 'L'
                  else
                        begin
                        mostrarerror(4);  exit;
                        end;                            //si <> 0 el directo-
rio recibido no es coherente.
                  end;
      end;
      analizarCD(dirBase);
End;
```

```
procedure analizarPWD(var ENTRADA: string);
{
   Nombre: analizarPWD.

}
var tipo: byte;
Begin
      if strOutPut = '' then tipo := 0 else tipo := 1;      // Tipo 0: Salida es-
tándar, tipo 1: Redirigir la salida.
      if (ENTRADA[0] >= #4) and not (flagInPut) then
            begin
            mostrarerror(2);
            exit;
            end
      else pwd(tipo);
End;


procedure analizarReDIR (var ENTRADA: string);
var   comando: string;
      i,j: word;
      ok: boolean;
      salida,respaldo:text;
      salidaRuta:string;
Begin
      i:= pos(' > ',ENTRADA);
      if i <> 0 then
            begin // > rewrite (reescribe el archivo, lo crea si no existe).
                  salidaRuta:= copy(ENTRADA,i+3,length(ENTRADA));
                  comando:= copy(ENTRADA,1,i-1);
                  crearArchivo(salida,salidaRuta,ok);
                  if not(ok) then
                        begin
                        mostrarerror(11);
                        exit;
                        end;
                  respaldarSalidaEstandar(respaldo);
                  redirigirSalidaEstandar(salida);
                  analizar(comando);
                  close(salida);
                  restaurarStdOut(respaldo);
            end
      else
      begin
            j:= pos(' >> ', ENTRADA);
            if j <> 0 then
                  begin // >> append (añade los datos al final del archivo).
                        salidaRuta:= copy(ENTRADA,j+4,length(ENTRADA));
                        comando:= copy(ENTRADA,1,j-1);
                        agregarArchivo(salida,salidaRuta,ok);
                        if not ok then
                              begin
                              Mostrarerror(12);
                              exit;
                              end;
                        respaldarSalidaEstandar(respaldo);
                        redirigirSalidaEstandar(salida);
                        analizar(comando);
                        restaurarStdOut(respaldo);
```

```
                end;
        end;
End;


procedure analizarPipe(var ENTRADA: string);
var i: word;
      preString, postString: string;
      respaldo,commandPipe:text;
      ok:boolean;
Begin
      i:= pos(' | ',ENTRADA);
      preString   := copy(ENTRADA, 1, i-1);
      postString  := copy(ENTRADA, i+3, length(ENTRADA));
      crearArchivo(commandPipe,'commandPipe',ok);
            if not(ok) then
                  begin
                  mostrarerror(13);
                  exit;
                  end;
      popen (commandPipe,postString,'W');
      respaldarSalidaEstandar(respaldo);
      redirigirSalidaEstandar(commandPipe);
      analizar(preString);
      restaurarStdOut(respaldo);
      Pclose(commandPipe);
      deletefile('commandPipe');
End;

procedure analizarBG(var ENTRADA: string);
var i:word;
Begin
      alBG:= true;
      i:=pos(' &',ENTRADA);
      ENTRADA:= copy(ENTRADA,1,i-1);
      analizar(ENTRADA);
end;
END.
```